Enterprise System Applications

Tutorial 6

Prof. Dr. Gamal Kassem

Habiba Nabil

Ali Suleiman

Ahmed Maged

Table of Content

Listings

1 Introduction

2 Screen 200

2.1 Editing Top Include

The first thing, as previously mentioned in the previous tutorial, is to edit the TOP include code for it to initialize the tables and data involved in the table control element.

To do that, first navigate to your package in module SE80 → expand programs → expand the program you have created in the previous tutorial → double-click on the TOP include → click on the pencil icon to edit the include.

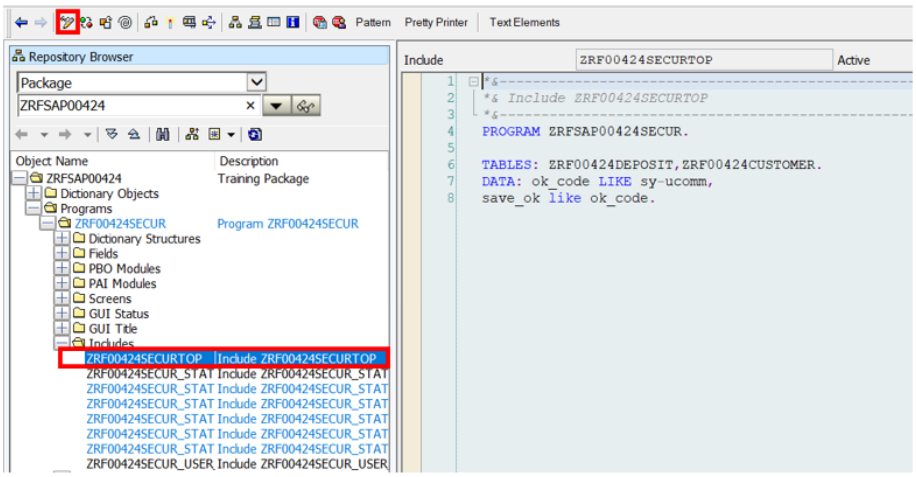

Figure 1: Edit TOP Include

We will add the following code to the include:

```
DATA:it_customernnn24 LIKE TABLE OF zvcustomernnn24,
    wa_customernnn24 LIKE zvcustomernnn24.
TABLES: zvcustomernnn24.
CONTROLS tc TYPE TABLEVIEW USING SCREEN '0200'.
DATA: mark
TYPE c, lines_it TYPE i, lines_tc TYPE i,
     lines_max TYPE i, pos TYPE i.
```

Make sure to pres *CTRL + F* and replace the ”nnn” value with your assigned user number.

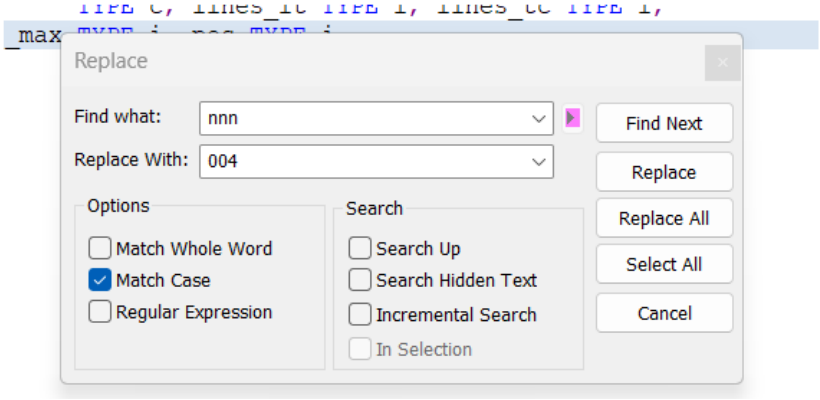

Figure 2: Replace nnn

Make sure to click replace all. Your code should look similar to the figure below:

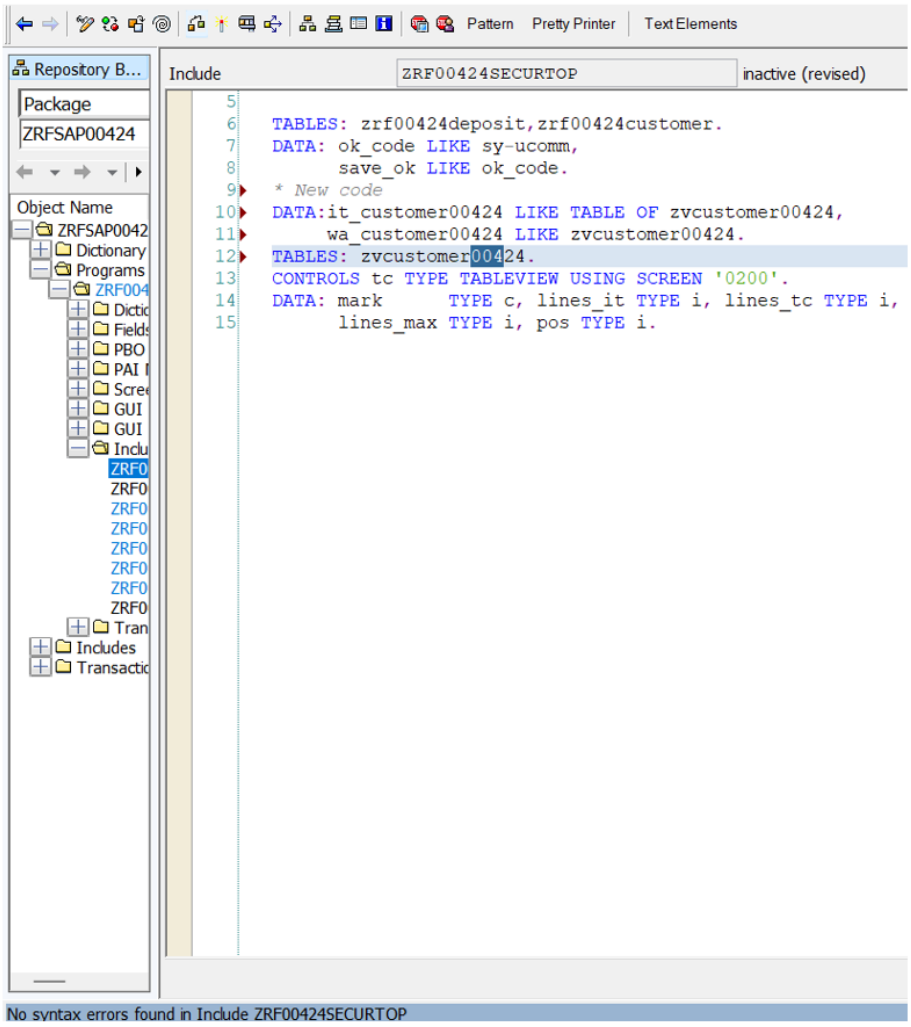

Figure 3: Final Code

2.2 Screen 200 Screen Painter

Now that we have defined the TOP include, we want to get back to the Layout editor to add our table control element.

Expand the screens package, double-click Screen 200 to open it and press on Layout

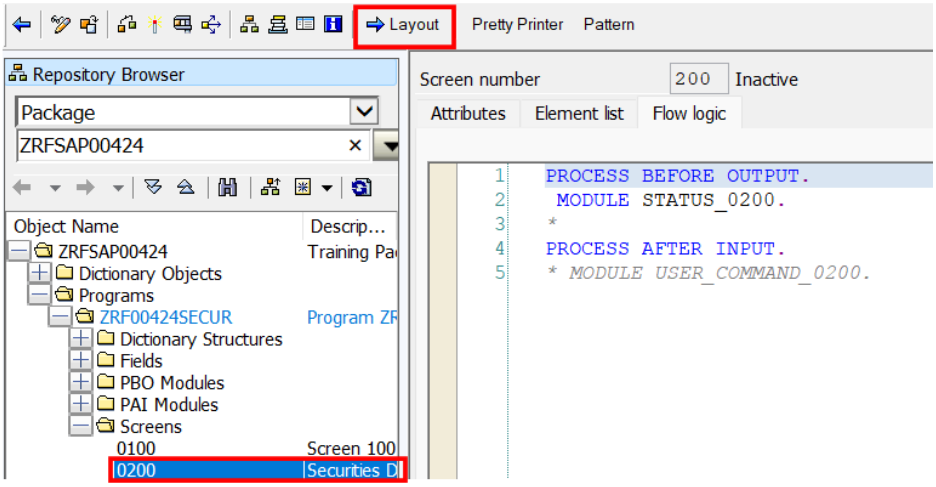

Figure 4: Open Layout Editor

Select the table control element. Instead of just clicking on the screen to add it, you should press and hold the left mouse button and drag on the screen to size the element appropriately.

Make sure that the name provided for the element is ”TC”.

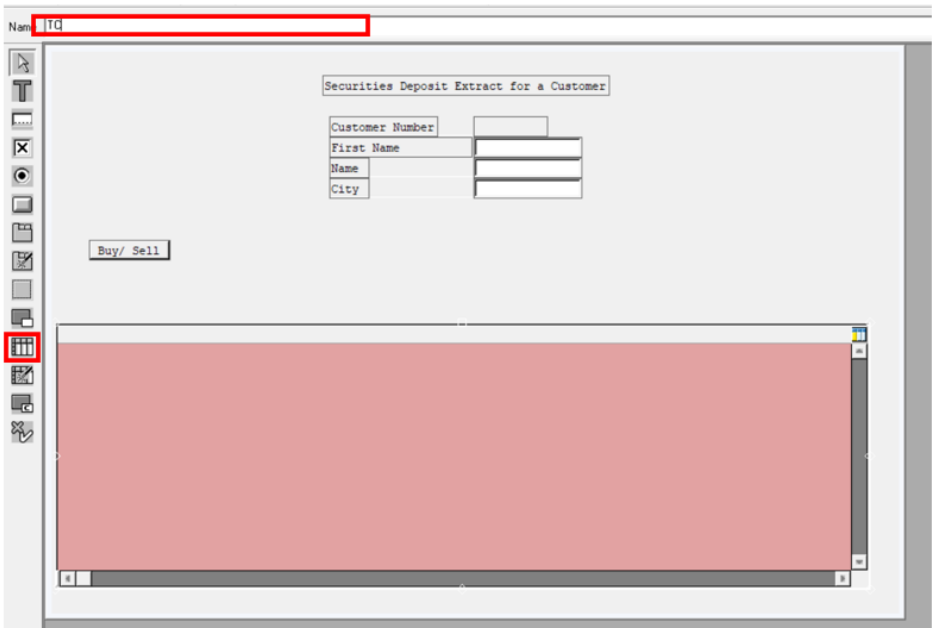

Figure 5: Insert Table Control Element

After that, we want to select the fields which the table control element will view, to do the definition you should click on dict./ program fields button.

As previously mentioned, we will feed the table control data from the database view table which we have created in the previous tutorial. In the Table/ Field Name type ”ZVCUSTOMERnnn24” and press ”Get From Dictionary”.

We will select the following data from the table:

- CODE
- TITLE
- EMITTER
- DEPOSITNR
- AMOUNT

Click on the green tick and place the fields inside the table control element.

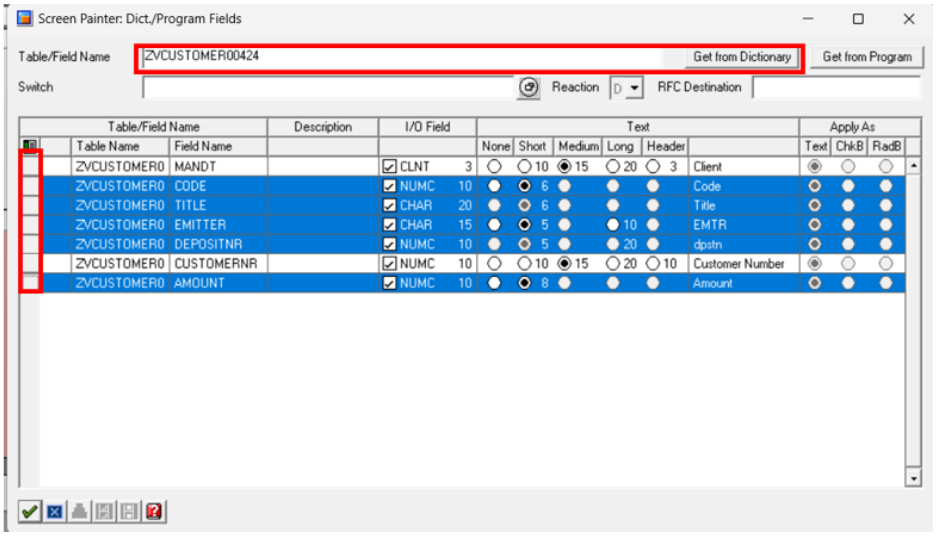

Figure 6: Select View Table

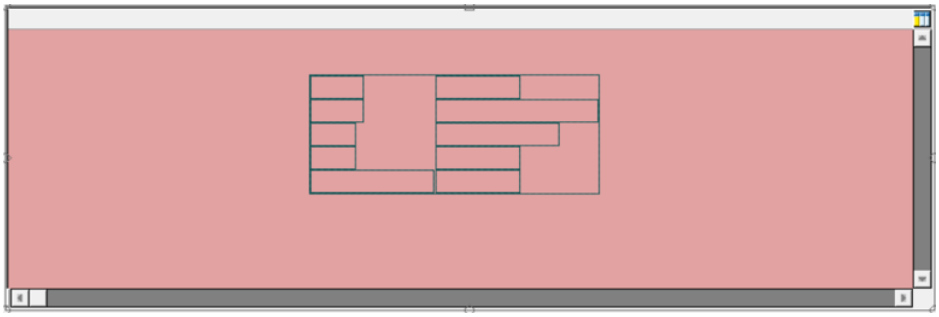

Figure 7: Adding Dictionary Fields to TC

Figure 8: TC Fields

Following

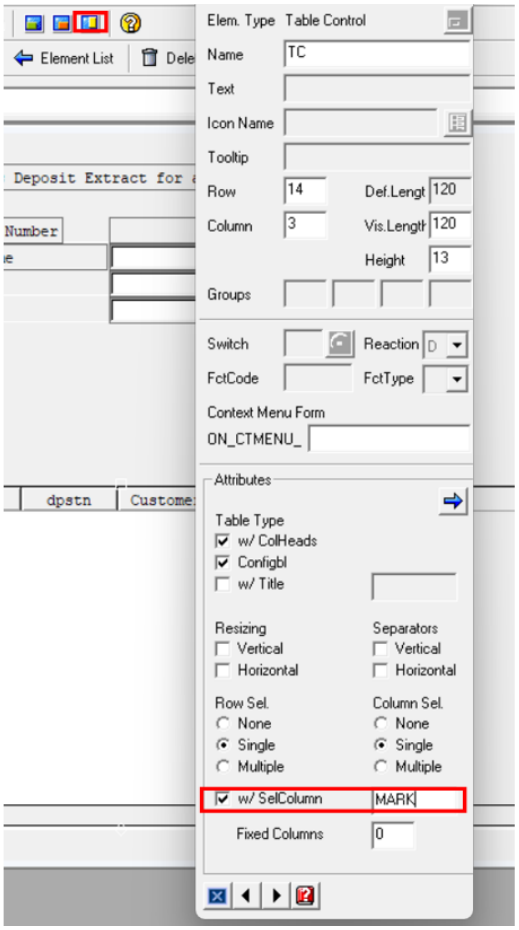

Figure 9: TC Attribute

Following that we want to make sure that the fields of the table control are read only. This is to ensure that the user won't change or alter the information in the database. To do that, double-click on one of the fields, in the attribute window select program. Click on output only. Repeat this step for the other fields.

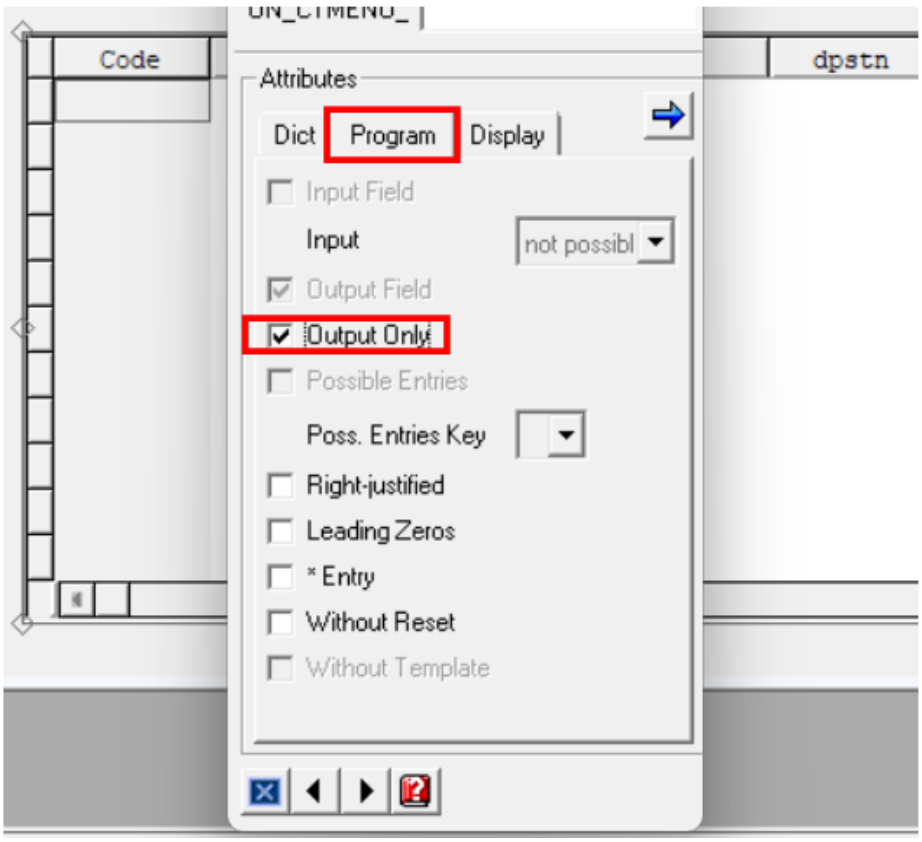

Figure 10: Making Fields Output Only

We have already defined the button and its function code in the previous tutorial, thus, we have completed all the elements needed for our screen 200 front-end. The only missing element is the ok code. Simply click on the element list button and type *ok_code* and press enter.

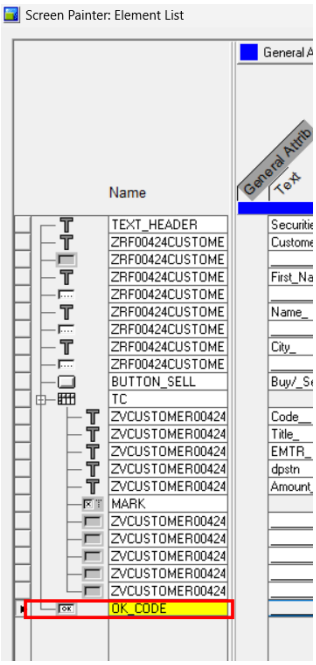

Figure 11: Ok Code

This is what your final screen should look like. If you followed all the steps correctly, simply save the screen and exit the screen painter.

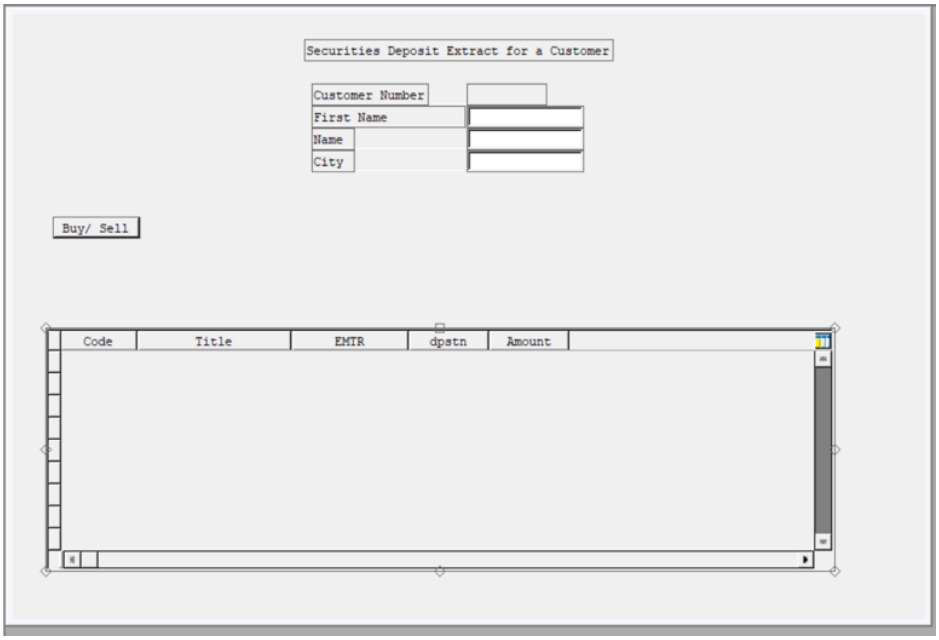

Figure 12: Final Screen 200

2.3 Module Transport

To create the module, navigate to the flow logic of screen 200. You should find it by default after exiting the screen painter. In the `PROCESS BEFORE OUTPUT` section, add the following code:

```
MODULE transport_to_0200.
```

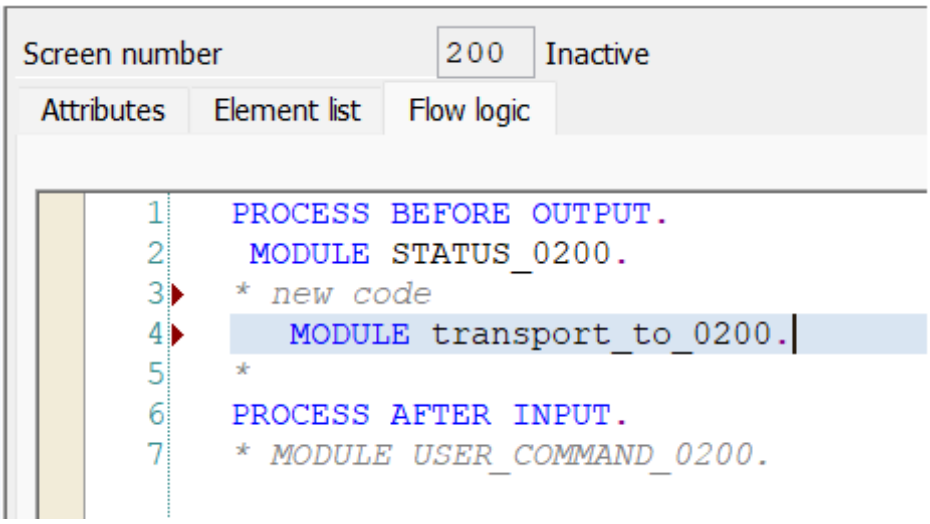

Figure 13: Adding Code to PBO

After writing the code, double-click "`transport_to_0200`" to create the module. You will be prompted to create the model and to save changes made to the flow logic. Click on "Yes" to both prompts.

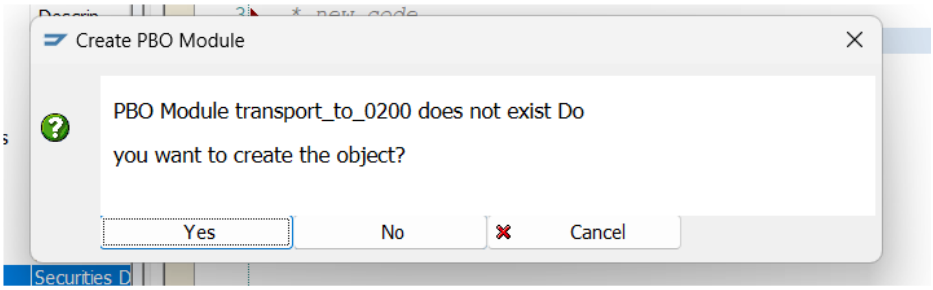

Figure 14: Creating New Module

Make sure, as always, to save the module to a new include.

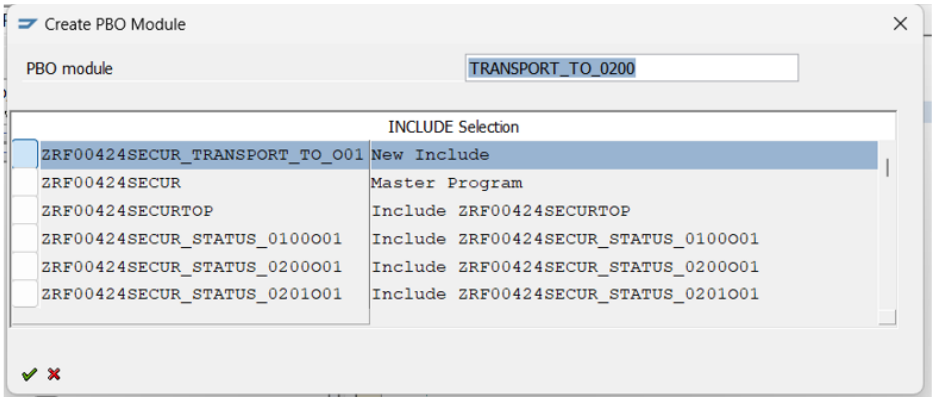

Figure 15: New Include

In the screen flow logic, add the following code to the module:

```
MODULE transport_to_0200 OUTPUT.
  SELECT SINGLE * FROM ZRFnnn24customer
  INTO ZRFnnn24customer
  WHERE customernr = ZRFnnn24deposit-customernr.
  CLEAR: mark, pos.
  DESCRIBE TABLE it_customernnn24 LINES lines_it.
  tc-lines = lines_it.
ENDMODULE.
```

Make sure to change the ”nnn” value to your assigned user number. Do this using the *CTRL* + *F* command.

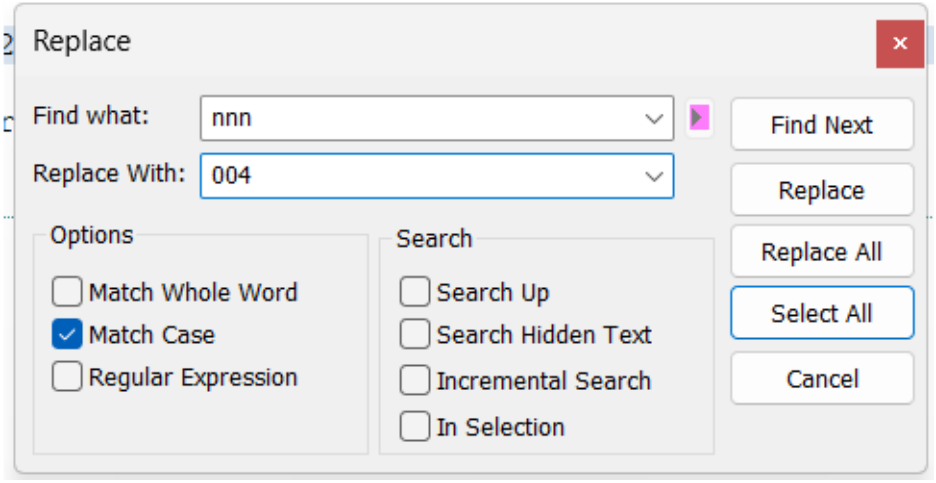

Figure 16: Replace nnn

This should be the final code for the module.

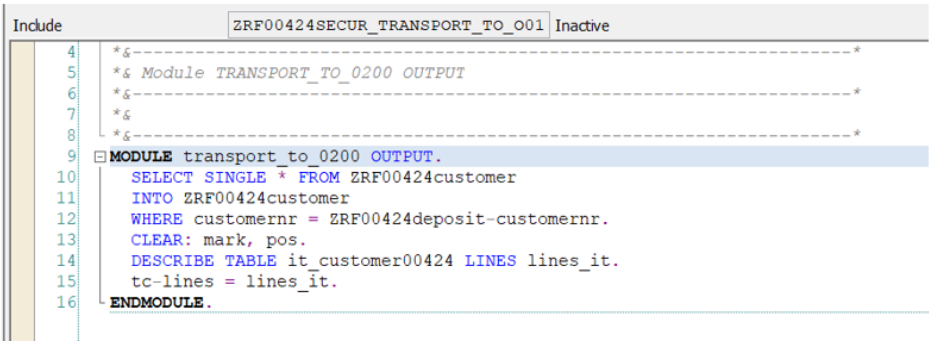

Figure 17: Final 'Transport' Code

2.4 Module tc_fill

The next module we will create is the `tc_fill` module. This module is responsible for filling the table control element with the data from the database view table. To do this, we will have to loop through the table and fill the table control element with the data.

This is achieved using the following code:

```
LOOP WITH CONTROL TC.
    MODULE tc_fill.
ENDLOOP.
```

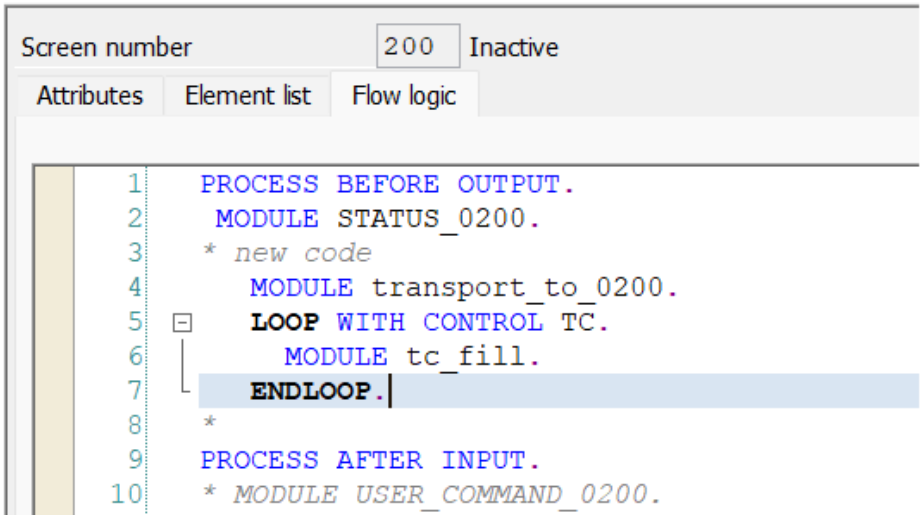

Figure 18: Loop Code

Double-click on `tc_fill` to create the module. You will be prompted to save the changes made to the flow logic. Click on "Yes". Make sure you create the module in a *new include*.

Add the following code to the module:

```
MODULE tc_fill OUTPUT.
    READ TABLE it_customernnn24 INTO zvcustomernnn24
    INDEX tc-current_line.
ENDMODULE.
```

Again, make sure to replace "nnn" with your assigned user number.

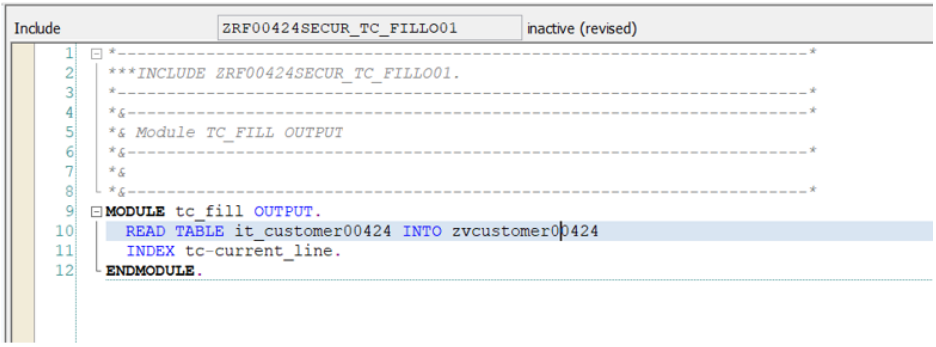

Figure 19: TC Fill Code

2.5 PAI Screen 100

Now that we defined the table control element and its logic, we can complete and finalize screen 100 PAI. When the user select a customer from screen 100, he should be able to see the deposits of that customer in screen 200. The deposit information will be displayed in the table control element.

Open the flow logic of screen 100 then, double-click `USER_COMMAND_0100`.

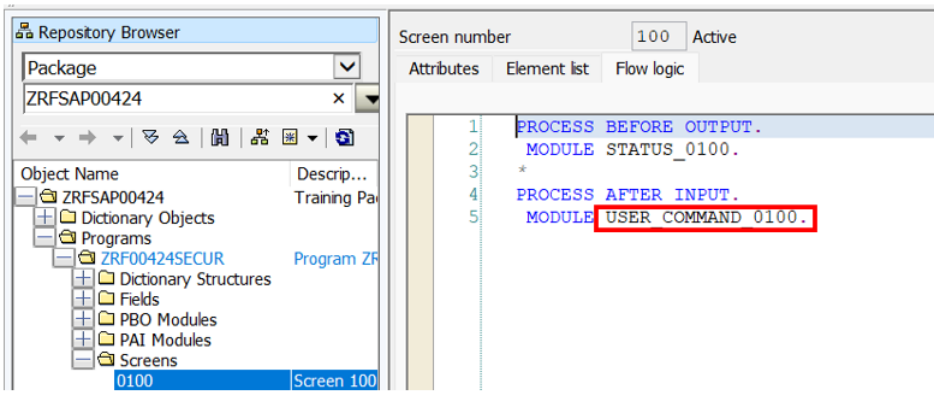

Figure 20: Screen 100 PAI

We want to add the following code after `WHEN 'SELECT'`:

```
SELECT * FROM zvcustomernnn24
INTO CORRESPONDING FIELDS OF TABLE it_customernnn24
WHERE customernr = ZRFnnn24deposit-customernr.
LEAVE TO SCREEN 200.
```

The code selects the customer information inserting them into the internal table `it_customernnn24`. The customer number is the key to the selection. After that, the code leaves the screen to screen 200.

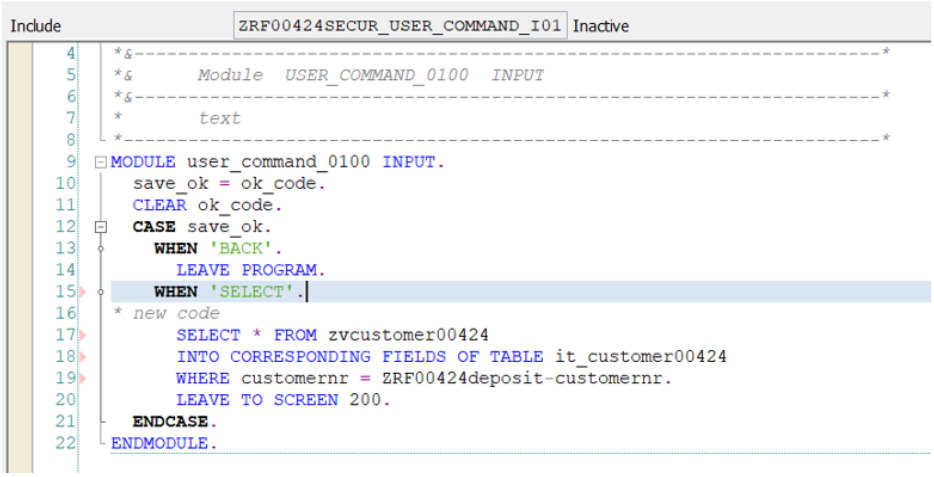

Figure 21: When Select Code

2.6 Screen 200 PAI

Now we are done with everyhting that has to do with the PBO of screen 200. We will now move to the PAI of screen 200.

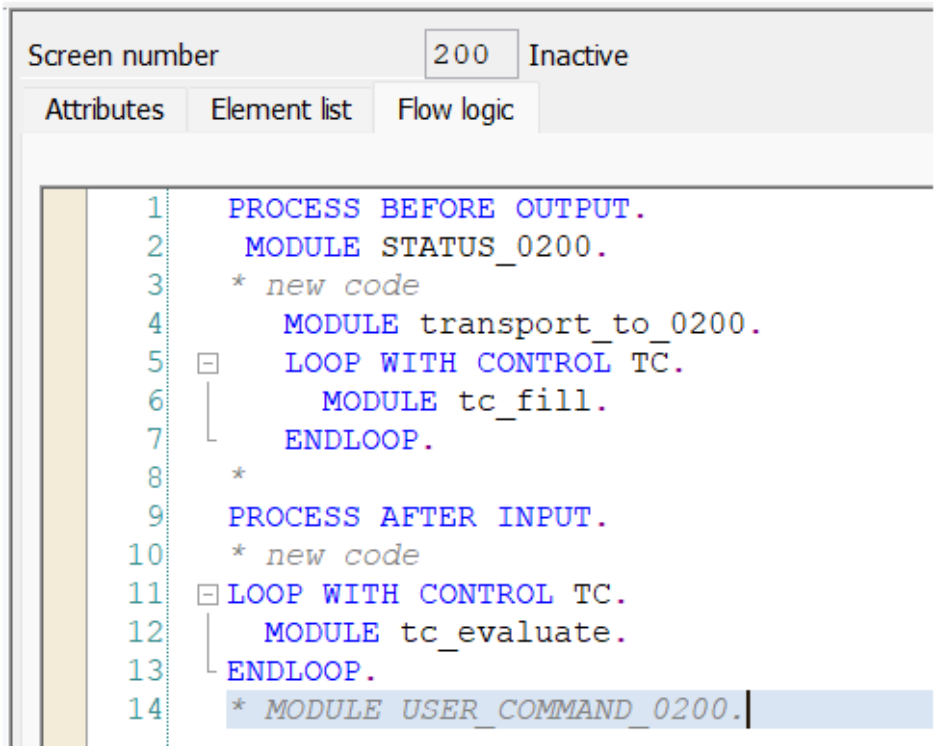

Figure 22: PAI Loop Code

```
LOOP WITH CONTROL TC.
    MODULE tc_evaluate.
ENDLOOP.
```

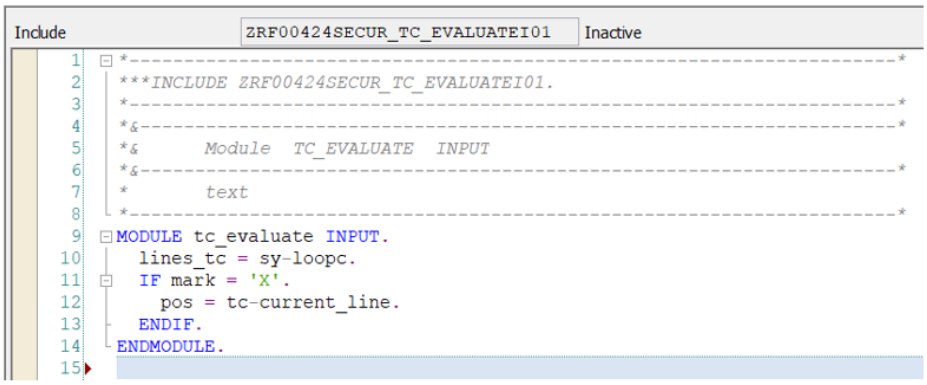

Figure 23: TC Evaluate Code

```
MODULE tc_evaluate INPUT.
    lines_tc = sy-loopc.
    IF mark = 'X'.
        pos = tc-current_line.
    ENDIF.
ENDMODULE.
```

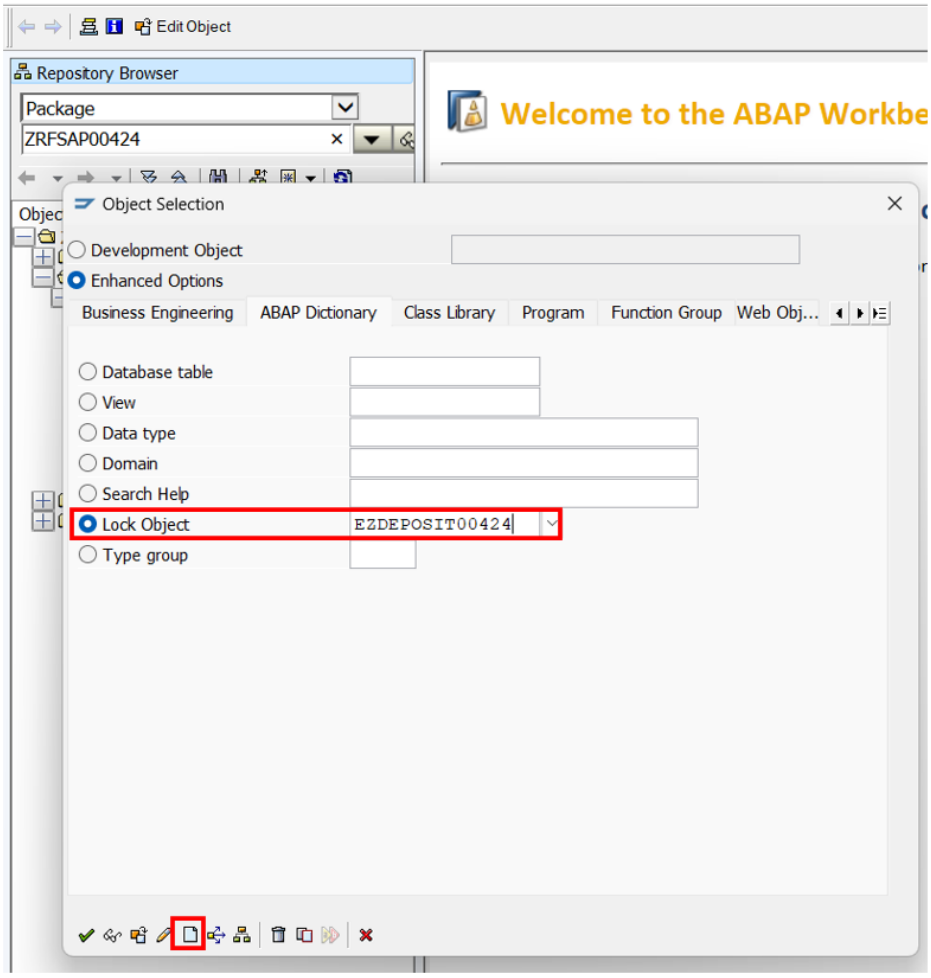

Figure 24: Lock Object

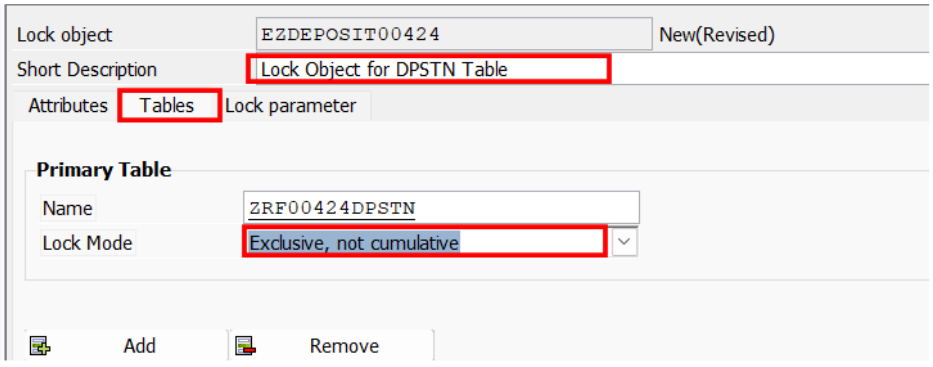

Figure 25: Lock Object Attributes

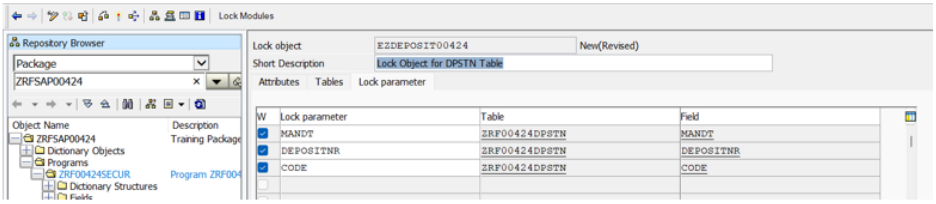

Figure 26: Lock Object Table

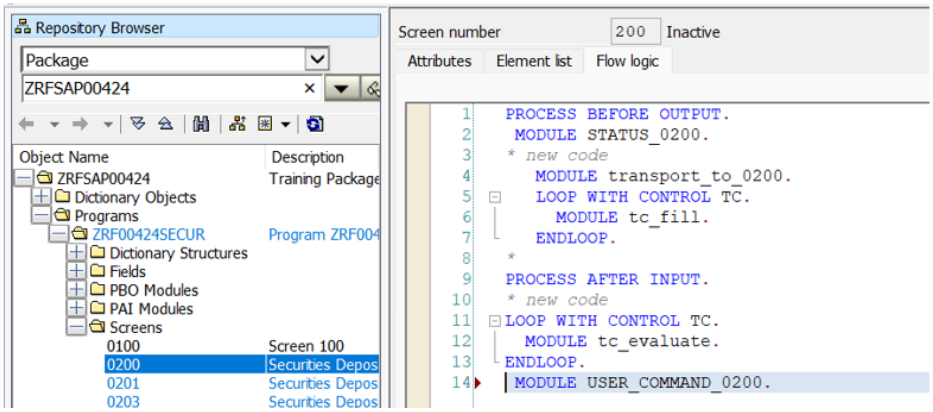

Figure 27: PAI Screen 200

```
MODULE USER_COMMAND_0200.
```

Include ZRF00424SECUR_USER_COMMAND_I02 inactive (revis

```
8   *---------------------------------------------------------------------*
9   MODULE user_command_0200 INPUT.
10    save_ok = ok_code.
11    CLEAR ok_code.
12    CASE save_ok.
13      WHEN 'BACK'.
14        SET SCREEN '0100'.
15      WHEN 'NEXT'.
16        tc-top_line = tc-top_line + lines_tc.
17        lines_max = lines_it - lines_tc + 1.
18        IF tc-top_line > lines_max.
19          tc-top_line = lines_max.
20        ENDIF.
21      WHEN 'PREV'.
22        tc-top_line = tc-top_line - lines_tc.
23        IF tc-top_line < 1.
24          tc-top_line = 1.
25        ENDIF.
26      WHEN 'LAST'.
27        tc-top_line =  lines_it - lines_tc + 1.
28      WHEN 'FIRST'.
29        tc-top_line = 1.
30      WHEN 'BUY'.
31        AUTHORITY-CHECK OBJECT 'Z_DPST00424'
32          ID 'ACTVT' FIELD '02'
33          ID 'ANZAHL' DUMMY.
34        IF sy-subrc NE 0.
35          MESSAGE e000(ZRF00424MESSAGE) WITH
36            'No authorization to perform this operation!'.
37        ELSE.
38          READ TABLE it_customer00424 INTO wa_customer00424 INDEX pos.
39          IF sy-subrc NE 0.
40            MESSAGE i000(ZRF00424message) WITH
41             'Select please valid table line!!!'.
42          ELSE.
43            SELECT SINGLE * FROM zvcustomer00424
44            INTO wa_customer00424
45            WHERE code = wa_customer00424-code
46               AND depositnr = wa_customer00424-depositnr.
47            IF sy-subrc NE 0.
48              MESSAGE i001(ZRF00424message) WITH
49              'Security ' wa_customer00424-code ' is deleted!'.
50            ELSE.
51              LEAVE TO SCREEN 300.
52            ENDIF.
53          ENDIF.
54        ENDIF.
55    ENDCASE.
56  ENDMODULE.
```

Figure 28: User Command 200 Code

```
MODULE user_command_0200 INPUT.
  save_ok = ok_code.
  CLEAR ok_code.
  CASE save_ok.
    WHEN 'BACK'.
      SET SCREEN '100'.
    WHEN 'NEXT'.
      tc-top_line = tc-top_line + lines_tc.
      lines_max = lines_it - lines_tc + 1.
      IF tc-top_line > lines_max.
        tc-top_line = lines_max.
      ENDIF.
    WHEN 'PREV'.
      tc-top_line = tc-top_line - lines_tc.
      IF tc-top_line < 1.
        tc-top_line = 1.
      ENDIF.
    WHEN 'LAST'.
      tc-top_line =  lines_it - lines_tc + 1.
    WHEN 'FIRST'.
      tc-top_line = 1.
    WHEN 'BUY'.
      AUTHORITY-CHECK OBJECT 'Z_DPSTnnn24'
        ID 'ACTVT' FIELD '02'
        ID 'ANZAHL' DUMMY.
      IF sy-subrc NE 0.
        MESSAGE e000(zrfnnn24message) WITH
          'No authorization to perform this operation!'.
      ELSE.
        READ TABLE it_customernnn24 INTO wa_customernnn24 INDEX pos.
        IF sy-subrc NE 0.
          MESSAGE i000(ZRFnnn24message) WITH
           'Select please valid table line!!!'.
        ELSE.
          SELECT SINGLE * FROM zvcustomernnn24
          INTO wa_customernnn24
          WHERE code = wa_customernnn24-code
             AND depositnr = wa_customernnn24-depositnr.
          IF sy-subrc NE 0.
            MESSAGE i001(ZRFnnn24message) WITH
            'Security ' wa_customernnn24-code ' is deleted!'.
          ELSE.
*           Lock Object code
            LEAVE TO SCREEN 300.
          ENDIF.
        ENDIF.
      ENDIF.
  ENDCASE.
ENDMODULE.
```

```
          ELSE.
*           Lock Object code
            CALL FUNCTION 'ENQUEUE_EZRF00424DPSTN'
              EXPORTING
                mode_zrf00424dpstn = 'X'
                mandt              = sy-mandt
                depositnr          = wa_customer00424-depositnr
                code               = wa_customer00424-code
*               X_DEPOSITNR        = ' '
*               X_CODE             = ' '
*               _SCOPE             = '2'
*               _WAIT              = ' '
*               _COLLECT           = ' '
              EXCEPTIONS
                foreign_lock       = 1
                system_failure     = 2
                OTHERS             = 3.
            IF sy-subrc <> 0.
* Implement suitable error handling here
              MESSAGE i000(zrf00424message) WITH
    'The position is locked by another user!'
     'Try again later!'.

            ENDIF.

            LEAVE TO SCREEN 300.
          ENDIF.
        ENDIF.
      ENDIF.
```

Figure 29: Enqueue Code

```
CALL FUNCTION 'ENQUEUE_EZRFnnn24DPSTN'
              EXPORTING
                mode_zrfnnn24dpstn = 'X'
                mandt              = sy-mandt
                depositnr          = wa_customernnn24-depositnr
                code               = wa_customernnn24-code
*               X_DEPOSITNR        = ' '
*               X_CODE             = ' '
*               _SCOPE             = '2'
*               _WAIT              = ' '
*               _COLLECT           = ' '
              EXCEPTIONS
                foreign_lock       = 1
                system_failure     = 2
                OTHERS             = 3.
            IF sy-subrc <> 0.
* Implement suitable error handling here
              MESSAGE i000(zrfnnn24message) WITH
              'The position is locked by another user!'
              'Try again later!'.
            ENDIF.
```

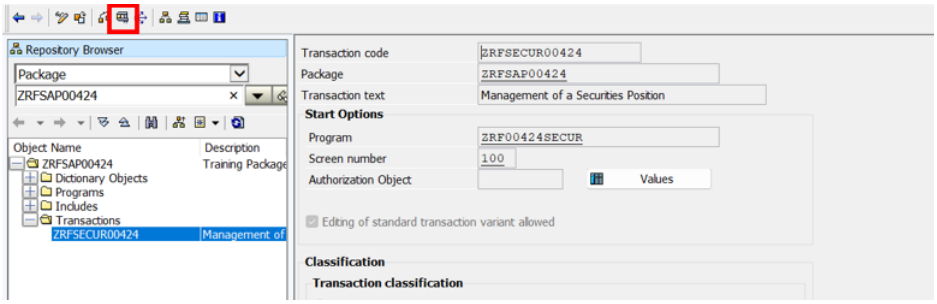

Figure 30: Testing